目录

概述

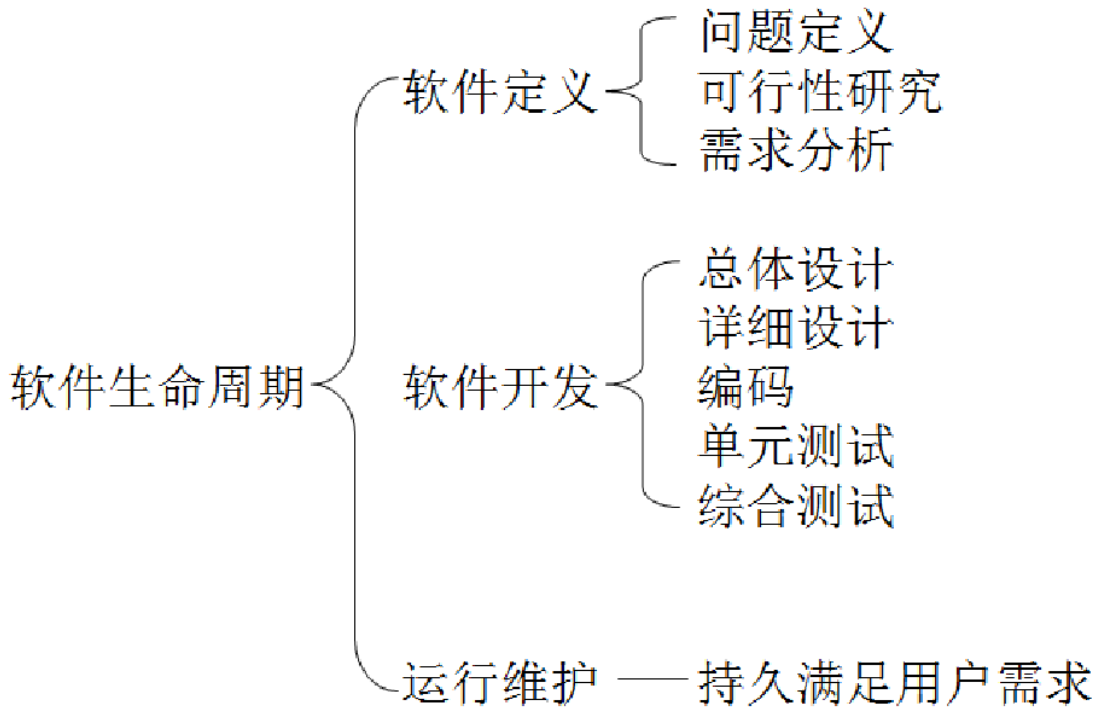
软件工程的出现是由于**软件危机**的出现。

软件工程方法学涉及的3要素：**工具**、**方法**、**过程**

管理与技术，对软件项目成功开发的作用与影响如何

> **管理是影响软件项目成功开发的全局性因素，而技术只影响局部**

***重点*：软件生命周期由软件定义、软件开发和运行维护（软件维护）三个时期组成，每个时期又进一步划分成若干个阶段，最后阶段是什么。**



螺旋模型将**瀑布模型、原型模型和增量模型**结合起来，加入了**风险分析。**

快速原型模型适用于什么情况：**需求不是很明确、需求较模糊等情况**

可行性研究

***重点*：可行性研究的目的是什么？应从哪些方面来考虑软件项目的可行性?**

目的：**不是解决问题**，而是确定问题**是否能够解决**、是否**值得去解决。**

考虑方面：

技术可行性：使用现有的技术能实现这个系统吗?

经济可行性：这个系统的经济效益能超过它的开发成本吗?

操作可行性：系统的操作方式在这个用户组织内行得通吗?

其它：法律、社会文化可行性等

可行性研究阶段，**数据流图**与**数据字典**共同构成系统的**逻辑模型**

数据字典的概念：

数据字典是关于数据的信息的集合，也就是对数据流图中包含的所有元素的定义的集合。

数据字典所定义的元素：

1. 数据流
2. 数据元素（数据流分量）
3. 数据存储
4. 处理。

需求分析

需求分析阶段的**基本任务**：

不是确定系统**怎样完成**它的工作，而是**确定**系统**必须完成哪些**工作

也就是对目标系统**提出完整、准确、清晰、具体的要求。**

ER图的三大**基本要素**：

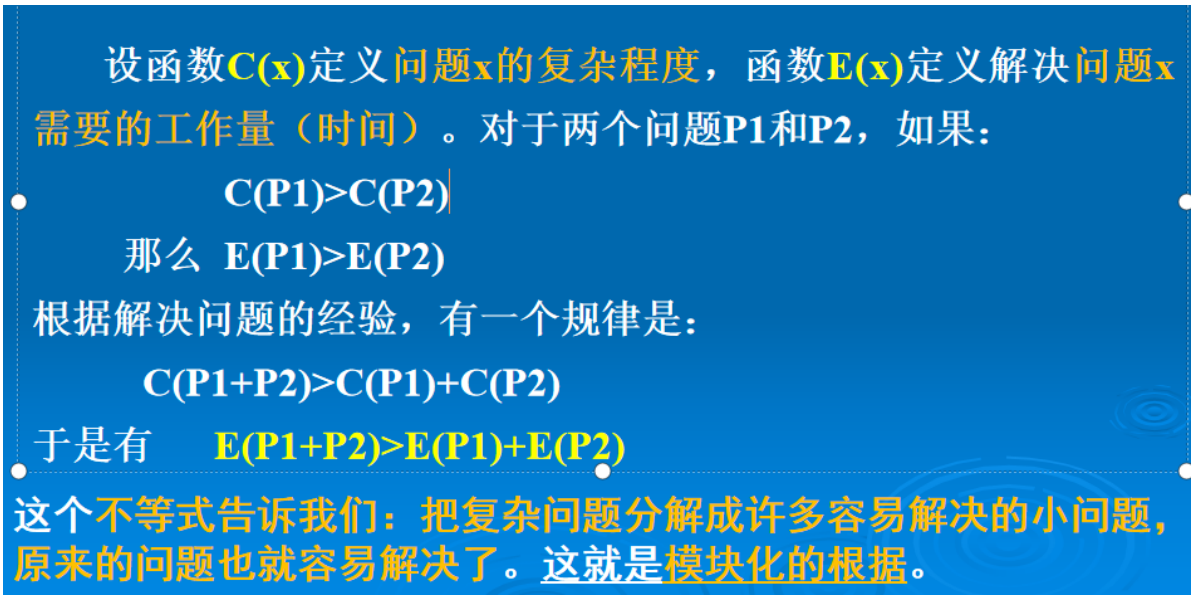
实体，属性，关系

数据库**物理设计**的**主要任务**是**设计并构建表和字段**

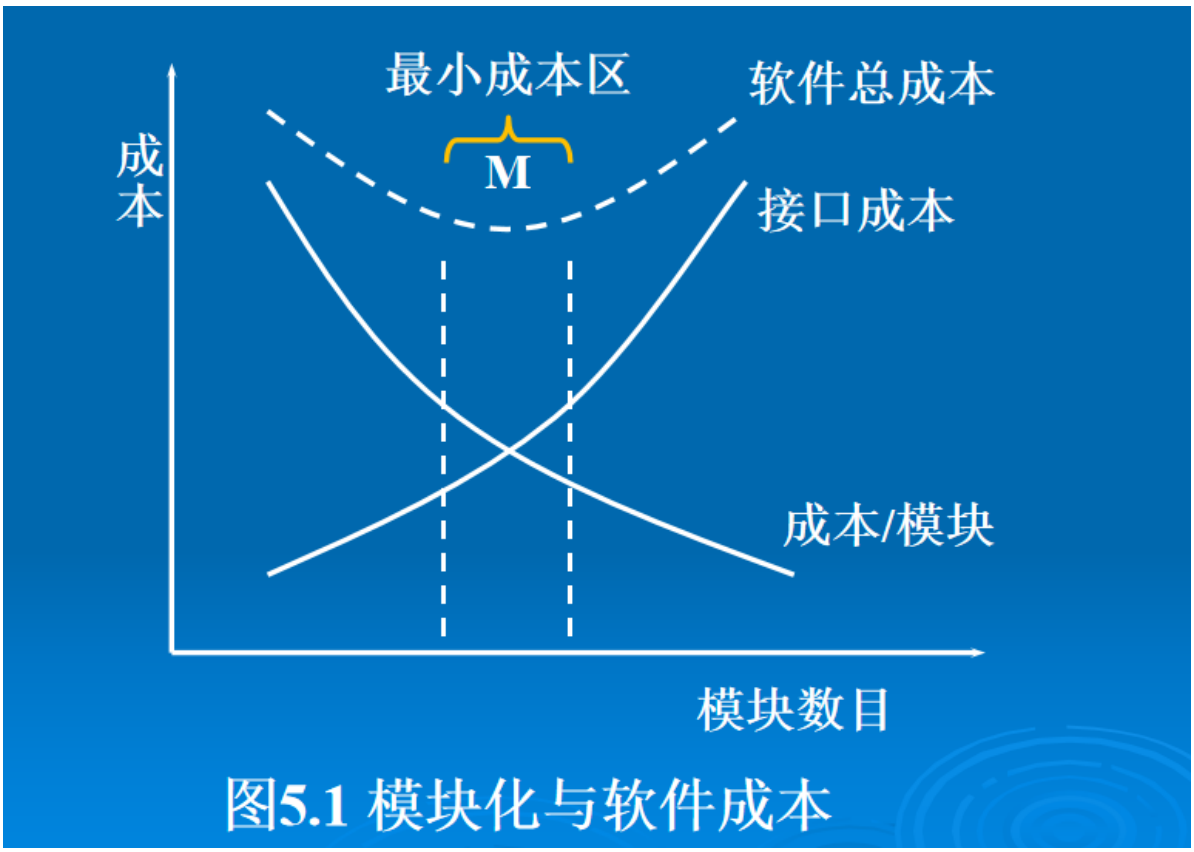
需求分析建模过程中， 应建立**数据模型**、**功能模型**和**行为模型**三种模型。

总体设计

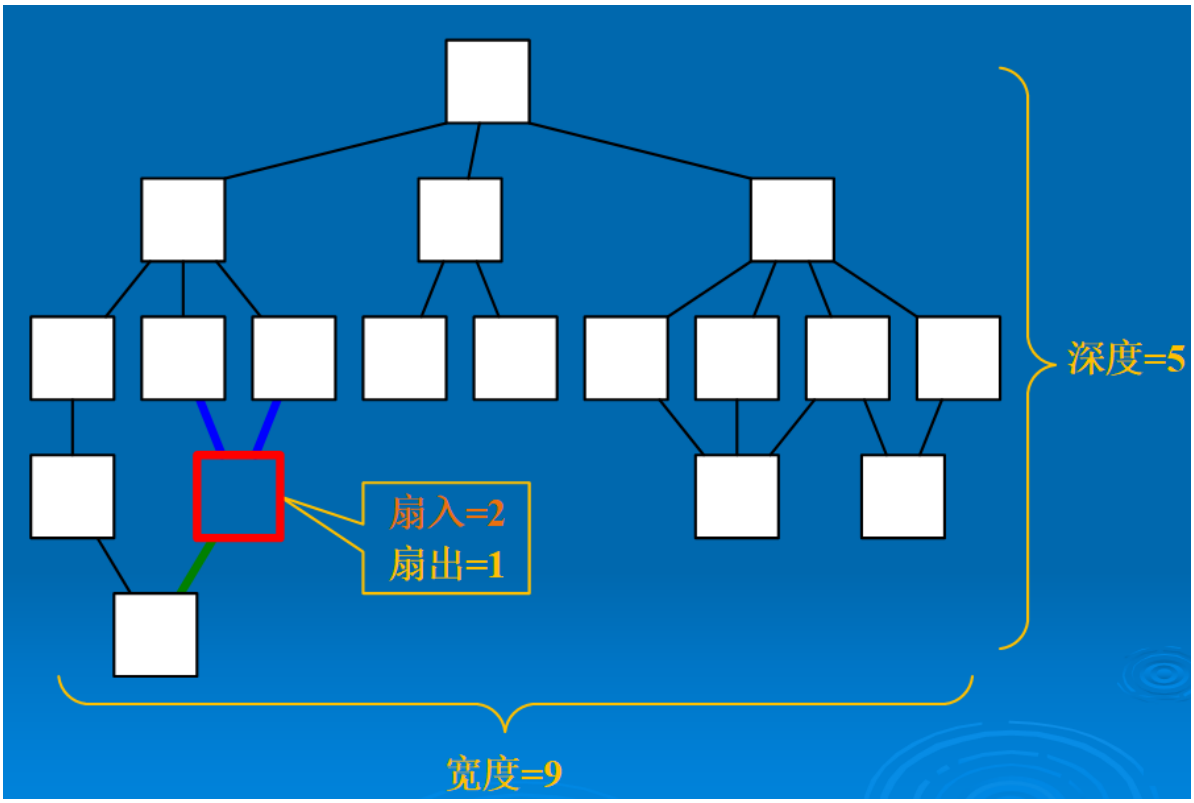
模块化的**根据**：



模块化（模块划分）与**软件成本**的关系



***重点，大题*：计算软件结构的深度、宽度、扇入、扇出**



- 深度：软件结构中控制的层数；
- 宽度：软件结构内同一个层次上的模块总数的最大值；
- 扇出：一个模块直接控制（调用）其它模块的数目；
- 扇入：一个模块被其它模块调用的数目。

耦合与**内聚**概念，软件模块的**独立程度**可以由这两个**定性**标准来度量。

耦合：指软件结构内**不同模块彼此之间**相互依赖（连接）的紧密程度。

内聚：一个**模块内部各个元素彼此结合**的紧密程度。

耦合针对模块间，内聚针对模块内部，追求**低耦合**，**高内聚**

模块内的**高内聚**往往意味着模块间的**低耦合**

数据流分析：包括**变换型**与**事务型**

变换型：变换型数据流的特征是可以把它看成由输入、**变换中心**和输出三部分组成

变换中心可以理解为数据的加工和处理程序

事务型：数据沿输入通路到达一个处理T，这个处理**根据输入数据的类型在若干个动作序列中选出一个来执行。**T称为**事务中心**

***重点*：启发式规则的7个方面：**

1. 改进软件结构提高模块独立性
2. 模块规模应该适中
3. 深度、宽度、扇出和扇入都应适当
4. 模块的作用域应该在控制域之内
5. 降低模块接口的复杂度
6. 设计单入口、单出口的模块
7. 模块功能应该可以预测

详细设计

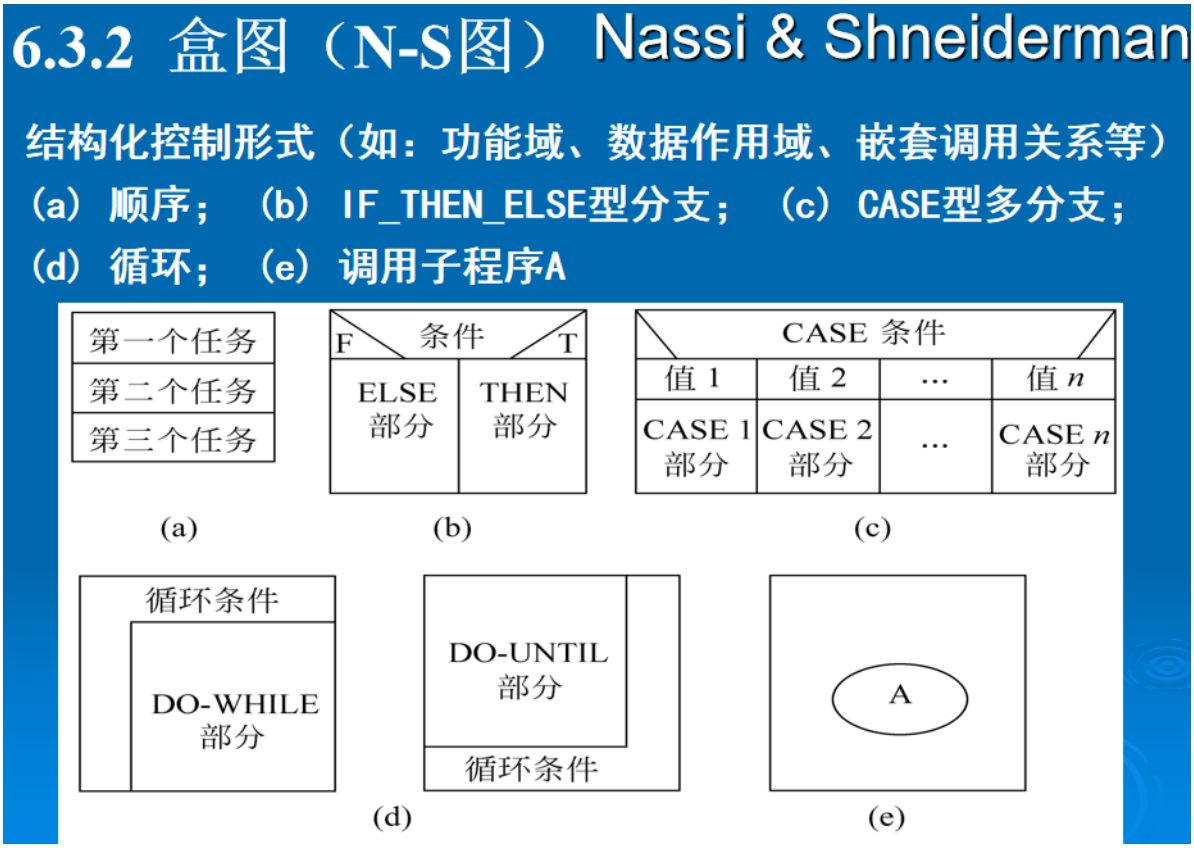
详细设计的**基本任务**是确定**每个模块的算法设计**

***重点*：详细设计的具体任务：**

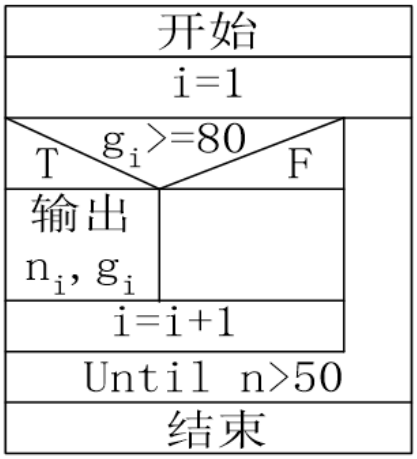
1. 算法设计
2. 数据结构设计
3. 模块接口细节
4. 测试用例设计
5. 数据库物理设计
6. 数据代码设计
7. 其他设计
8. 编写详细设计说明书并进行评审。

程序的三种**基本控制结构**是**顺序**、**选择**和**循环**

***重点，大题*：用N-S图（盒图）、PAD图表示程序控制过程**



例：用N-S图表示:打印50名学生中成绩在80分及以上者的学号和成绩。



N-S图

基本类型

(a) 顺序(先执行P1后执行P2)；

(b) 选择(IF C THEN P1 ELSE P2)；

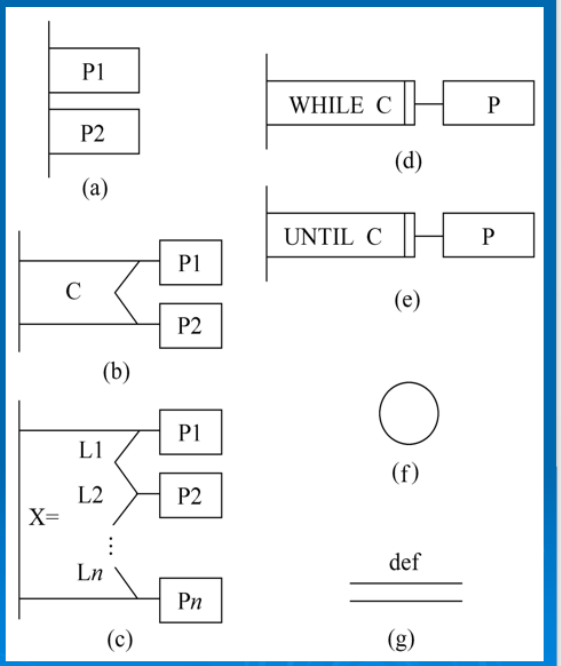
(c) CASE型多分支；

(d) WHILE型循环(WHILE C DO P)；

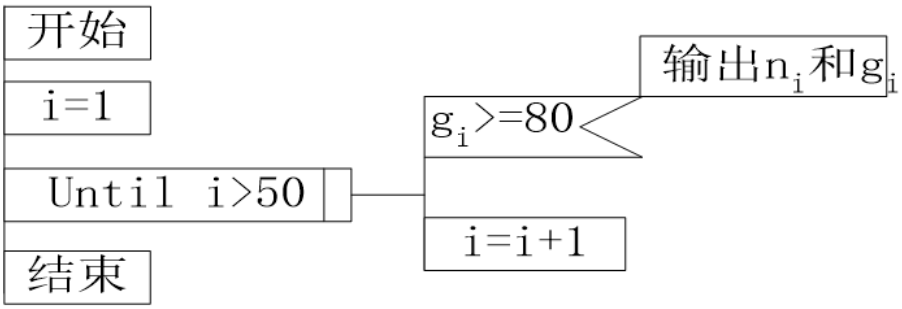
(e) UNTIL型循环(REPEAT P UNTIL C)；

(f) 语句标号；

(g) 定义



例：用PAD图表示：打印50名学生中成绩在80分及以上者的学号和成绩。



PAD图

软件设计中，通过**抽象**和**分解**可以**降低复杂性。**

PDL是一种什么语言：

过程设计语言，也被称之为**伪代码**

***重点*：给张判定表，对其解释**

表 6.1　用判定表表示计算行李费的算法

一张判定表由4部分组成：

左上部列出所有条件

右上部表示各种条件组合的一个矩阵

左下部是所有可能做的动作

右下部是和每种条件组合相对应的动作。

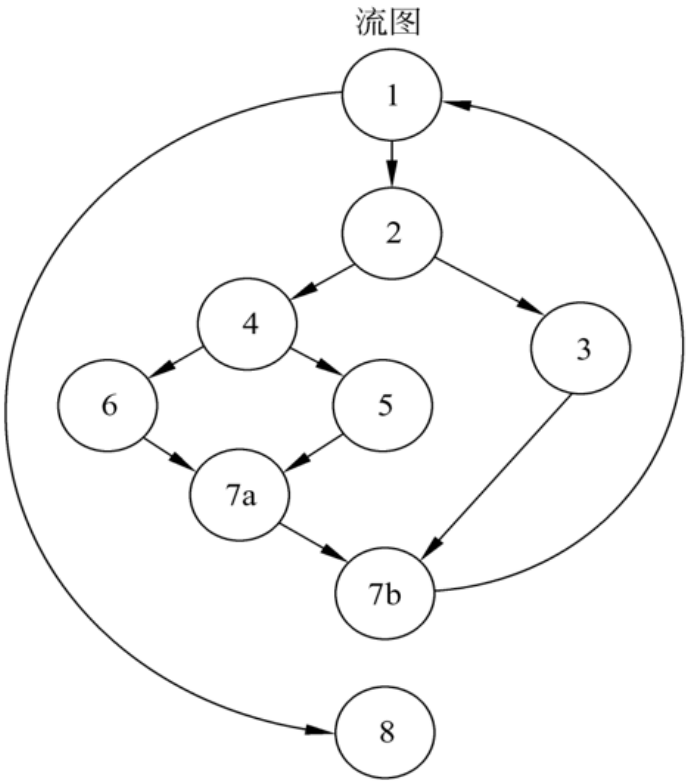
	规则								
	1	2	3	4	5	6	7	8	9
国内乘客		T	T	T	T	F	F	F	F
头等舱		T	F	T	F	T	F	T	F
残疾乘客		F	F	T	T	F	F	T	T
行李重量 $W \leqslant 30$kg	T	F	F	F	F	F	F	F	F
免费	×								
$(W-30)\times 2$				×					
$(W-30)\times 3$					×				
$(W-30)\times 4$		×						×	
$(W-30)\times 6$			×						×
$(W-30)\times 8$						×			
$(W-30)\times 12$							×		

•判定表右半部的每一列实质上是一条规则，规定了与特定的条件组合相对应的动作。

当算法中包含**多重嵌套**的**条件选择**时，用程序流程图、盒图、PAD图等都不易清楚地描述，然而，**判定表**能够清晰地表示**复杂的条件组合**与**应做的动作**之间的**对应关系。**

***重点，大题*：PDL转化为流图的方法，环形复杂度的计算方法**

```
PDL
procedure:sort
1:    do while records remain
2:        read record;
          if record field 1=0
3:          then process record;
              store in buffer;
              incremert counter;
4:        elseif record field 2=0
5:          then reset counter;
6:          else process record;
              store in file;
7a:       endif
          endif
7b:   enddo
8:    end
```



执行顺序的多个语句合并成一个节点，节点中包含：选择语句|选择的分支|选择的汇总

环形复杂度计算：

1. 环形复杂度 V（G）等于流图中的**区域数**；
2. 环形复杂度 **V（G） = E - N+2**，其中E是流图中**边的条数**，N是**结点数**；
3. 环形复杂度 V（G） = P + 1，其中P为流图中判定结点的数目。

软件实现

对于涉及**用户量巨大、要求极高**的软件产品，其**测试人员数量必然比开发人员数量多。**

测试**只能证明程序有错误**，不能证明程序没有错误。

软件测试中，**确认（验收）测试**主要用于发现**需求分析阶段**的错误。

边界值分析法的测试方式：

边界值分析不是从某等价类中随便挑一个作为代表，而是使这个**等价类的每个边界**都要作为测试条件。

选取刚好**等于、稍大于、稍小于**等价类边界值的测试数据：

软件可靠性的内涵：

软件**可靠性**是程序在给定的**时间间隔**内，按照规格说明书的规定**成功地运行的概率**

软件**可用性**是程序在给定的**时间点**，按照规格说明书的规定**成功地运行的概率。**

白盒测试也叫**逻辑覆盖测试**，包括：**语句覆盖**，**判定覆盖**，**条件覆盖。**

语句覆盖：程序中每个语句至少执行一次

判定覆盖：每个判定的真假分支都至少执行一一次

条件覆盖：判定表达式的每个条件都取到各种可能的结果

条件覆盖通常比判定覆盖强，但条件覆盖不一定包含判定覆盖。

黑盒测试有**等价类法**和**边界值分析法。**

常在测试过程的**早期阶段**主要使用**白盒测试**，而在**后期阶段**主要使用**黑盒测试。**

软件测试的三大或四大阶段:

1. 模块测试（单元测试）：把每个模块作为单独的实体来测试
2. 子系统测试（集成测试）：把经过单元测试的模块放在一起形成一个子系统来测试，着重测试模块的接口
3. 系统测试 (集成测试)：把经过测试的子系统装配成一个完整的系统来测试
4. 验收测试（确认测试）：把软件系统作为单一的实体进行测试（利用用户的实际数据测试）

软件调试途径主要有以下3种：**蛮干法**、**回溯法**和**原因排除法。**

软件维护

软件维护是软件**生命周期**中所**花费用最多**的阶段

对于**非结构化维护**，其**软件配置**的**唯一成分**是**程序代码**；而**结构化维护**则有一个**完整的软件配置**存在。

软件维护包括**改正性维护**、**完善性维护**、**适应性维护**和**预防性维护**四种。

术语翻译

DFD图 ： 数据流图

描绘信息流和数据从输入移动到输出的过程中所经受的变换

E-R图：实体-联系图

用来建立数据模型的工具

IPO图：输入、处理、输出图

美国IBM公司发展完善起来的一种图形工具，能够方便的描绘输入数据、对数据的处理和输出数据之间的关系

HIPO图：层次图 + 输入/处理/输出图

HIPO图中的每个方框需要对应一张IPO图或IPO表

PAD图：问题分析图 Problem Analysis Diagram

用二维树形结构的图来表示程序的控制流

PDL：过程设计语言 Process Design Language

也称为伪码

UML：统一建模语言 *Unified Modeling Language*

用来对软件密集系统进行可视化建模的一种语言

CASE：结构化分析与设计工具

OOP：面向对象编程 *Object Oriented Programming*

把OOD的设计结果转为代码

OOD：面向对象设计

把我们的领域模型转为逻辑架构，类图，类之间的关系

SWEBOK：软件工程知识体系

- 软件工程 Software Engineering
- 软件需求 Software Requirements
- 软件设计 Software Design
- 软件过程 Software Process
- 开源项目 Open source project
- 人工智能　artificial intelligence
- 软件部署 software deployment
- 软件建构 Software Construction
- 系统软件 System software
- 应用软件 Application software
- 嵌入式软件 Embedded software
- 分布式计算 Distributed computing
- 软件过程 Software Process
- 软件测试 Software Test
- 软件质量 Software Quality
- 软件危机 Software Crisis